EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S14	180	(replac$4 substitut$4) with ((machine adj code) (machine adj language))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/15 15:07
S33	2	(detect$4 identif$4 determin$4) with ((single adj instruction adj multiple adj data) (SIMD)) same (intermediate adj language)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/15 17:46
S34	1	(detect$4 identif$4 determin$4) with ((single adj instruction adj multiple adj data) (SIMD)) same (object adj code)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/15 17:47
S36	7	(header adj file) with source adj (code program) same (intermediate adj (code language))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/16 17:47
S37	91	((SIMD) (single adj instruction adj multiple adj data)) with class	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/17 11:30
S38	29	((SIMD) (single adj instruction adj multiple adj data)) with class and compil$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 11:43
S39	10	((SIMD) (single adj instruction adj multiple adj data)) with ((intermediate adj code) (intermediate adj language) (object adj code))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/17 12:03
S41	140	(information data) with processor with machine with language and compiler	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:38

EAST Search History

S42	66	function with invocation with compiler	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:40
S43	3	intermediate with (code language) with function adj invocation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:42
S44	12	intermediate with (code language) same function adj invocation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:42
S45	37	(substitut$4 replac$4) same ((intermediate adj code) (intermediate adj language) (object adj code)) same ((machine adj language) (machine adj code))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:42
S46	31	((SIMD) (single adj instruction adj multiple adj data)) with class and compil$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/28 15:45
S47	8	compil$4 with (intermediate adj code) same function with invocation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 11:27
S48	5	((intermediate adj code) (intermediate adj language)) with function with invocation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 11:27
S49	29	(substitut$4 replac$4) with ((intermediate adj code) (intermediate adj language) (object adj code)) same ((machine adj language) (machine adj code))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 11:42

EAST Search History

S50	1	(class adj library) same (replac$4 substitut$4) with (machine adj language)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 11:42
S51	31	((SIMD) (single adj instruction adj multiple adj data)) with class and compil$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 12:03
S58	26	compil$4 same pars$4 same optimiz$4 same intermediate and simd	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 12:07
S59	41	header adj file and (((machine adj program) (machine adj code) (machine adj language)) same ((intermediate adj code) (object adj code) (language adj processor)))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 12:33
S60	2	"5758164".pn.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/01/02 12:34

Sign in

Web Images Video News Maps **more »**

function invocation compiler Search Advanced Search Preferences

Web Results **1** - **10** of about **621,000** for **function invocation compiler**. (**0.27** seconds)

Function invocation...
Can any body plz tell me, the rules followed by the **compiler**(gcc) when it is invoking the **function**. Bcoz, in my program(MPICH-G2), when a **compiler** has to ...
www-unix.globus.org/mail_archive/mpich-g/2005/06/msg00017.html - 5k -
Cached - Similar pages

[20] Inheritance virtual **functions**, C++ FAQ Lite
The **compiler** uses the static type of the pointer to determine whether the member **function invocation** is legal. If the type of the pointer can handle the ...
www.cacs.louisiana.edu/~mgr/404/burks/language/cpp/cppfaq/virtualf.htm - 16k -
Cached - Similar pages

[Globus-discuss] **Function invocation...**
[Globus-discuss] **Function invocation**. ... Bcoz, in my program(MPICH-G2), when a **compiler** has to invoke a **function** - instead of that, it is terminating and ...
www.globus.org/mail_archive/discuss/2005/06/msg00506.html - 5k -
Cached - Similar pages

C++ vs C Performance Comparison (Virtual **Functions**, Inheritance ...
The C++ **compiler** inserts an extra pointer to a vtable which ... Virtual **Function Invocation**, Virtual **function invocation** is slightly more expensive than ...
www.eventhelix.com/RealtimeMantra/Basics/ComparingCPPAndCPerformance2.htm - 41k -
Cached - Similar pages

Compiler-friendly programming
Compiler-friendly programming idioms can be as useful to performance as any of ... features are costly in object space and **function invocation** performance. ...
publib.boulder.ibm.com/.../topic/com.ibm.vacpp6m.doc/
getstart/overview/opt_cmp_friendly_programming.htm - 5k - Cached - Similar pages

Legion 1.6 Developer Manual: 3.0 Mentat
In Mentat, the **compiler** and run-time system detect that the first two ... between independent objects is accomplished via member **function invocation**. ...
legion.virginia.edu/documentation/tutorials/1.6/manuals/Developer_1_6.5.html - 15k -
Cached - Similar pages

[20] Inheritance -- virtual **functions**, C++ FAQ Lite
Static typing means that the legality of a member **function invocation** is checked at the earliest possible moment: by the **compiler** at compile time. ...
geneura.ugr.es/~jmerelo/c++-faq/virtual-**functions**.html - 31k - Cached - Similar pages

[PS] Integration of Message Passing into an IDL **Compiler**
File Format: Adobe PostScript - View as Text
only synchronous **function invocation**. I introduce. a more general approach to communication { mes- ... the **compiler** generates receive and wait C **functions** ...
i30www.ira.uka.de/~kevinelp/workshop/papers/Aigner.ps - Similar pages

Version 9 of the Icon **Compiler**
Debugging, string **invocation**, linking code from other Icon programs, and external **functions** are handled somewhat differently in the **compiler** from the way ...
www.cs.arizona.edu/icon/docs/ipd237.htm - 18k - Cached - Similar pages

25.1.3. Compilation Environment
The **compiler** must treat as a **function** call any form that is a list whose car is a ... is

executed and possibly as late as generic **function invocation** time. ...
www.supelec.fr/docs/cltl/clm/node227.html - 17k - Cached - Similar pages

Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

Download Google Pack: free essential software for your PC

function invocation compiler Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

PORTAL USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ◉ The ACM Digital Library ○ The Guide

+function +invocation +compiler SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **function** **invocation** **compiler**

Found **3,638** of **193,448**

Sort results by	relevance	Save results to a Binder	Try an Advanced Search
Display results	expanded form	Search Tips	Try this search in The ACM Guide
		☐ Open results in a new window	

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Context-sensitive interprocedural points-to analysis in the presence of function pointers

Maryam Emami, Rakesh Ghiya, Laurie J. Hendren

June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6

Publisher: ACM Press

Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper reports on the design, implementation, and empirical results of a new method for dealing with the aliasing problem in C. The method is based on approximating the points-to relationships between accessible stack locations, and can be used to generate alias pairs, or used directly for other analyses and transformations. Our method provides context-sensitive interprocedural information based on analysis over invocation graphs that capture all calling contexts including re ...

2 Commutativity analysis: a new analysis technique for parallelizing compilers

Pedro C. Diniz

November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6

Publisher: ACM Press

Full text available: pdf(472.62 KB) Additional Information: full citation, abstract, references, citings, index terms

This article presents a new analysis technique, commutativity analysis, for automatically parallelizing computations that manipulate dynamic, pointer-based data structures. Commutativity analysis views the computation as composed of operations on objects. It then analyzes the program at this granularity to discover when operations commute (i.e., generate the same final result regardless of the order in which they execute). If all of the operations required to perform a given computation com ...

Keywords: parallel computing

3 Handling context-sensitive syntactic issues in the design of a front-end for a MATLAB compiler

Pramod G. Joisha, Abhay Kanhere, Prithviraj Banerjee, U. Nagaraj Shenoy, Alok Choudhary

March 2001 **ACM SIGAPL APL Quote Quad**, Volume 31 Issue 3

Publisher: ACM Press

Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references

In recent times, the MATLAB language has emerged as a popular alternative for programming in diverse application domains such as signal processing and meteorology. The language has a powerful array syntax with a large set of pre-defined operators and functions that operate on arrays or array sections, making it an ideal candidate for applications involving substantial array-based processing.Yet, for all the programming convenience that the language offers, designing a parser and scanner capable ...

Keywords: assignments, colon expressions, command-form function invocations, control constructs, matrices, single quote character, syntax analysis for MATLAB

4 Technical Correspondence: On the return types of virtual functions
Emanuele Panizzi, Bernardo Pastorelli
June 1999 **ACM SIGPLAN Notices**, Volume 34 Issue 6
Publisher: ACM Press
Full text available: pdf(429.47 KB) Additional Information: full citation, abstract, references

In this paper a problem raising in the static type checking of C++ virtual functions is presented. C++ allows a certain degree of freedom in the specialization of the return type of virtual functions. As to the actual draft standard, this freedom can lead to run-time errors and, for this reason, it must be used carefully or compiler support must be added to prevent inconsistencies.

Keywords: C++, static typing, subtyping, virtual functions

5 A reconfigurable dataflow machine for implementing functional programming languages
Christophe Giraud-Carrier
September 1994 **ACM SIGPLAN Notices**, Volume 29 Issue 9
Publisher: ACM Press
Full text available: pdf(771.29 KB) Additional Information: full citation, index terms

6 Portable run-time support for dynamic object-oriented parallel processing
Andrew S. Grimshaw, Jon B. Weissman, W. Timothy Strayer
May 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 2
Publisher: ACM Press
Full text available: pdf(2.21 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Mentat is an object-oriented parallel processing system designed to simplify the task of writing portable parallel programs for parallel machines and workstation networks. The Mentat compiler and run-time system work together to automatically manage the communication and synchronization between objects. The run-time system marshals member function arguments, schedules objects on processors, and dynamically constructs and executes large-grain data dependence graphs. In this article we presen ...

Keywords: MIMD, dataflow, distributed memory, object-oriented, parallel processing

7 An optimizing compiler for LISP for the Z80
Jed Marti
August 1982 **Proceedings of the 5th ACM SIGSMALL symposium on Small systems**
Publisher: ACM Press
Full text available: pdf(430.22 KB) Additional Information: full citation, abstract, references, index terms

This paper describes an optimizing compiler for the Z80. Described are the compilation mechanisms, optimization techniques, and performance statistics.

8 A recovery mechanism for modular software
Flaviu Cristian
September 1979 **Proceedings of the 4th international conference on Software engineering**
Publisher: IEEE Press
Full text available: pdf(897.65 KB) Additional Information: full citation, abstract, references, citings, index terms

When an exception occurs, the state of a system is damaged : further processing may cause additional exceptions. Under some hypotheses concerning the system structure we give an a priori estimate of the damage caused by exceptions. Based on this estimate we propose a recovery strategy and a recovery mechanism.

9 LCM: memory system support for parallel language implementation
James R. Larus, Brad Richards, Guhan Viswanathan
November 1994 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the sixth international conference on Architectural support for programming languages and operating systems ASPLOS-VI**, Volume 29 , 28 Issue 11 , 5
Publisher: ACM Press
Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

Higher-level parallel programming languages can be difficult to implement efficiently on parallel machines. This paper shows how a flexible, compiler-controlled memory system can help achieve good performance for language constructs that previously appeared too costly to be practical.Our compiler-controlled memory system is called Loosely Coherent Memory (LCM). It is an example of a larger class of Reconcilable Shared Memory (RSM) systems, which generalize the replication and mer ...

10 pHluid: the design of a parallel functional language implementation on workstations
Cormac Flanagan, Rishiyur S. Nikhil
June 1996 **ACM SIGPLAN Notices , Proceedings of the first ACM SIGPLAN international conference on Functional programming ICFP '96**, Volume 31 Issue 6
Publisher: ACM Press
Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the distributed memory implementation of a shared memory parallel functional language. The language is Id, an implicitly parallel, mostly functional language that is currently evolving into a dialect of Haskell. The target is a distributed memory machine, because we expect these to be the most widely available parallel platforms in the future. The difficult problem is to bridge the gap between the shared memory language model and the distributed memory machine model. The lan ...

Keywords: data flow, garbage collection and run-time systems, parallel and distributed implementations

11 Levels of abstraction and compilers
B. P. Buckles, G. C. Hintze
October 1976 **Proceedings of the annual conference**
Publisher: ACM Press
Full text available: pdf(469.94 KB) Additional Information: full citation, abstract, references, citings, index terms

Based upon experience gained through the development of a compiler, this paper recommends the software partitioning technique known as levels of abstraction as a practical strategy for organizing medium to large-scale software systems. How to identify levels of abstraction, specific properties possessed by levels of abstraction, and how to integrate the technique into the software design phase are the principal topics. Each concept is illustrated by examples from the compiler. Some parallel ...

12 Functioning without closure: type-safe customized function representations for standard ML
Allyn Dimock, Ian Westmacott, Robert Muller, Franklyn Turbak, J. B. Wells
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming ICFP '01**, Volume 36 Issue 10
Publisher: ACM Press
Full text available: pdf(257.35 KB) Additional Information: full citation, abstract, references, citings, index terms

The CIL compiler for core Standard ML compiles whole ML programs using a novel typed intermediate language that supports the generation of type-safe customized data representations. In this paper, we present empirical data comparing the relative efficacy of several different flow-based customization strategies for function representations. We develop a cost model to interpret dynamic counts of operations required for each strategy. In this cost model, customizing the representation of closed fun ...

13 Design and evaluation of dynamic optimizations for a Java just-in-time compiler
Toshio Suganuma, Toshiaki Yasue, Motohiro Kawahito, Hideaki Komatsu, Toshio Nakatani
July 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 4
Publisher: ACM Press
Full text available: pdf(1.60 MB) Additional Information: full citation, abstract, references, index terms

The high performance implementation of Java Virtual Machines (JVM) and Just-In-Time (JIT) compilers is directed toward employing a dynamic compilation system on the basis of online runtime profile information. The trade-off between the compilation overhead and performance benefit is a crucial issue for such a system. This article describes the design and implementation of a dynamic optimization framework in a production-level Java JIT compiler, together with two techniques for profile-directed o ...

Keywords: JIT compiler, Recompilation, adaptive optimization, code specialization, dynamic compilation, profile-directed method inlining

14 Functional languages in microcode compilers
S. J. Allan
August 1989 **ACM SIGMICRO Newsletter , Proceedings of the 22nd annual workshop on Microprogramming and microarchitecture MICRO 22**, Volume 20 Issue 3
Publisher: ACM Press
Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, index terms

This paper discusses the advantages of using high-level languages in the development of microcode. It also describes reasons functional programming languages should be considered as the source language for microcode compilers. The emergence of parallel execution in microarchitectures dictates that parallelism must be extracted from the microcode programs. This paper shows how functional languages meet the needs of microprogrammers by allowing them to express their algorithms in natural ways ...

15 Inline function expansion for compiling C programs
P. P. Chang, W.-W. Hwu
June 1989 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation PLDI '89**, Volume 24 Issue 7
Publisher: ACM Press
Full text available: pdf(1.14 MB) Additional Information: full citation, abstract, references, citings, index terms

Inline function expansion replaces a function call with the function body. With automatic inline function expansion, programs can be constructed with many small functions to handle complexity and then rely on the compilation to eliminate most of the function calls. Therefore, inline expansion serves a tool for satisfying two conflicting goals: minizing the complexity of the program development and minimizing the function call overhead of program execution. A simple inline expansion procedur ...

16 Using types to analyze and optimize object-oriented programs
Amer Diwan, Kathryn S. McKinley, J. Eliot B. Moss
January 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 1
Publisher: ACM Press
Full text available: pdf(414.51 KB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented programming languages provide many software engineering benefits, but these often come at a performance cost. Object-oriented programs make extensive use of method invocations and pointer dereferences, both of which are potentially costly on modern machines. We show how to use types to produce effective, yet simple, techniques that reduce the costs of these features in Modula-3, a statically typed, object-oriented language. Our compiler performs type-based alias analysis to ...

Keywords: alias analysis, classes and objects, method invocation, object orientation, polymorphism, redundancy elimination

17 CLOS: integrating object-oriented and functional programming
Richard P. Gabriel, Jon L. White, Daniel G. Bobrow
September 1991 **Communications of the ACM**, Volume 34 Issue 9
Publisher: ACM Press
Full text available: pdf(2.53 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Lisp has a long history as a functional language,* where action is invoked by calling a procedure, and where procedural abstraction and encapsulation provide convenient modularity boundaries. A number of attempts have been made to graft object-oriented programming into this framework without losing the essential character of Lisp—to include the benefits of data abstraction, extensible type classification, incremental operator definition, and code reuse through an ...

Keywords: Common Lisp Object Systems

18 Compilers I: Compiler support for efficient processing of XML datasets
Xiaogang Li, Renato Ferreira, Gagan Agrawal
June 2003 **Proceedings of the 17th annual international conference on Supercomputing**
Publisher: ACM Press
Full text available: pdf(189.03 KB) Additional Information: full citation, abstract, references, index terms

Declarative, high-level, and/or application-class specific languages are often successful in easing application development. In this paper, we report our experiences in compiling a recently developed XML Query Language, XQuery for applications that process scientific datasets.Though scientific data processing applications can be conveniently represented in XQuery, compiling them to achieve efficient execution involves a number of challenges. These are, 1) analysis of recursive functions to ident ...

Keywords: XML, XQuery, data intensive computing, restructing compilers

19 Caching function calls using precise dependencies
Allan Heydon, Roy Levin, Yuan Yu
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference**

on Programming language design and implementation PLDI '00, Volume 35 Issue 5
Publisher: ACM Press
Full text available: pdf(243.57 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the implementation of a purely functional programming language for building software systems. In this language, external tools like compilers and linkers are invoked by function calls. Because some function calls are extremely expensive, it is obviously important to reuse the results of previous function calls whenever possible. Caching a function call requires the language interpreter to record all values on which the function call depends. For optimal caching, it is i ...

20 Dynamic compilation techniques: Inlining java native calls at runtime
Levon Stepanian, Angela Demke Brown, Allan Kielstra, Gita Koblents, Kevin Stoodley
June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**
Publisher: ACM Press
Full text available: pdf(416.42 KB) Additional Information: full citation, abstract, references, index terms

We introduce a strategy for inlining native functions into Java™ applications using a JIT compiler. We perform further optimizations to transform inlined *callbacks* into semantically equivalent lightweight operations. We show that this strategy can substantially reduce the overhead of performing JNI calls, while preserving the key safety and portability properties of the JNI. Our work leverages the ability to store statically-generated IL alongside native binaries, to facilitate nati ...

Keywords: JIT compilation, JNI, Java, inlining, native code

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL

USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

+function +invocation +intermediate

SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **function** **invocation** **intermediate**

Found **2,293** of **193,448**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200

Best 200 shown

Result page: **1** 2 3 4 5 6 7 8 9 10 next

Relevance scale

1 Safe fusion of functional expressions

Wei-Ngan Chin

January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming LFP '92**, Volume V Issue 1

Publisher: ACM Press

Full text available: pdf(1.10 MB)

Additional Information: full citation, abstract, references, citings, index terms

Large functional programs are often constructed by decomposing each big task into smaller tasks which can be performed by simpler functions. This hierarchical style of developing programs has been found to improve programmers' productivity because smaller functions are easier to construct and reuse. However, programs written in this way tend to be less efficient. Unnecessary intermediate data structures may be created. More function invocations may be required.

2 Context-sensitive interprocedural points-to analysis in the presence of function pointers

Maryam Emami, Rakesh Ghiya, Laurie J. Hendren

June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6

Publisher: ACM Press

Full text available: pdf(1.74 MB)

Additional Information: full citation, abstract, references, citings, index terms

This paper reports on the design, implementation, and empirical results of a new method for dealing with the aliasing problem in C. The method is based on approximating the points-to relationships between accessible stack locations, and can be used to generate alias pairs, or used directly for other analyses and transformations. Our method provides context-sensitive interprocedural information based on analysis over invocation graphs that capture all calling contexts including re ...

3 A user-centred approach to functions in excel

Simon Peyton Jones, Alan Blackwell, Margaret Burnett

August 2003 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN international conference on Functional programming ICFP '03**, Volume 38 Issue 9

Publisher: ACM Press

Full text available: pdf(210.80 KB)

Additional Information: full citation, abstract, references, citings, index terms

We describe extensions to the Excel spreadsheet that integrate user-defined functions into the spreadsheet grid, rather than treating them as a "bolt-on". Our first objective was to bring the benefits of additional programming language features to a system that is often not recognised as a programming language. Second, in a project involving the evolution of a well-established language, compatibility with previous versions is a major issue, and maintaining this compatibility was our second objec ...

4 A type system for certified binaries
Zhong Shao, Valery Trifonov, Bratin Saha, Nikolaos Papaspyrou
January 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 1
Publisher: ACM Press
Full text available: pdf(477.48 KB) Additional Information: full citation, abstract, references, index terms

A *certified binary* is a value together with a proof that the value satisfies a given specification. Existing compilers that generate certified code have focused on simple memory and control-flow safety rather than more advanced properties. In this article, we present a general framework for explicitly representing complex propositions and proofs in typed intermediate and assembly languages. The new framework allows us to reason about certified programs that involve effects while still mai ...

Keywords: Certified code, proof-preserving compilation, typed intermediate languages

5 A type system for certified binaries
Zhong Shao, Bratin Saha, Valery Trifonov, Nikolaos Papaspyrou
January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '02**, Volume 37 Issue 1
Publisher: ACM Press
Full text available: pdf(287.94 KB) Additional Information: full citation, abstract, references, citings

A *certified binary* is a value together with a proof that the value satisfies a given specification. Existing compilers that generate certified code have focused on simple memory and control-flow safety rather than more advanced properties. In this paper, we present a general framework for explicitly representing complex propositions and proofs in typed intermediate and assembly languages. The new framework allows us to reason about certified programs that involve effects while still maint ...

6 Type-preserving compilation of Featherweight Java
Christopher League, Zhong Shao, Valery Trifonov
March 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 2
Publisher: ACM Press
Full text available: pdf(378.51 KB) Additional Information: full citation, abstract, references, citings, index terms

We present an efficient encoding of core Java constructs in a simple, implementable typed intermediate language. The encoding, after type erasure, has the same operational behavior as a standard implementation using vtables and self-application for method invocation. Classes inherit super-class methods with no overhead. We support mutually recursive classes while preserving separate compilation. Our strategy extends naturally to a significant subset of Java, including interfaces and privacy. The ...

Keywords: Java, object encodings, type systems, typed intermediate languages

7 Aspects of applicative programming for file systems (Preliminary Version)
Daniel P. Friedman, David S. Wise
March 1977 **ACM SIGSOFT Software Engineering Notes , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of an ACM conference on Language design for reliable software**, Volume 2 , 11 , 12 Issue 2 , 2 , 3

Publisher: ACM Press

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper develops the implications of recent results in semantics for applicative programming. Applying suspended evaluation (call-by-need) to the arguments of file construction functions results in an implicit synchronization of computation and output. The programmer need not participate in the determination of the pace and the extent of the evaluation of his program. Problems concerning multiple input and multiple output files are considered: typical behavior is illustrated with an exam ...

Keywords: Functional combination, Real time, Recursive programming, Referential transparency, Shared file, Suspension, Text editor

8 From system F to typed assembly language
Greg Morrisett, David Walker, Karl Crary, Neal Glew
May 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 3
Publisher: ACM Press

Full text available: pdf(483.91 KB) Additional Information: full citation, abstract, references, citings, index terms

We motivate the design of typed assembly language (TAL) and present a type-preserving ttranslation from Systemn F to TAL. The typed assembly language we pressent is based on a conventional RISC assembly language, but its static type sytem provides support for enforcing high-level language abstratctions, such as closures, tuples, and user-defined abstract data types. The type system ensures that well-typed programs cannot violatet these abstractionsl In addition, the typing constructs admit ...

Keywords: certified code, closure conversion, secure extensible systems, type-directed compilation, typed assembly language, typed intermediate languages

9 Using Java reflection to automate extension language parsing
Dale Parson
December 1999 **ACM SIGPLAN Notices , Proceedings of the 2nd conference on Domain-specific languages PLAN '99**, Volume 35 Issue 1
Publisher: ACM Press

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, index terms

An extension language is an interpreted programming language designed to be embedded in a domain-specific framework. The addition of domain-specific primitive operations to an embedded extension language transforms that vanilla extension language into a domain-specific language. The LUxWORKS processor simulator and debugger from Lucent uses Tcl as its extension language. After an overview of extension language embedding and LUxWORKS experience, this paper looks at using Java reflection and ...

10 Implementation and evaluation of a QoS-capable cluster-based IP router
Prashant Pradhan, Tzi-cker Chiueh
November 2002 **Proceedings of the 2002 ACM/IEEE conference on Supercomputing**
Publisher: IEEE Computer Society Press

Full text available: pdf(215.68 KB) Additional Information: full citation, abstract, references, index terms

A major challenge in Internet edge router design is to support both high packet forwarding performance and versatile and efficient packet processing capabilities. The thesis of this research project is that a cluster of PCs connected by a high speed system area network provides an effective hardware platform for building routers to be used at the edges of the Internet. This paper describes a scalable and extensible edge router architecture called *Panama*, which supports a novel aggregate r ...

11

UnicStep-a visual stepper for COMMON LISP: portability and language aspects
Ivo Haulsen, Angela Sodan
July 1989 **ACM SIGPLAN Lisp Pointers**, Volume III Issue 1
Publisher: ACM Press
Full text available: pdf(773.52 KB) Additional Information: full citation, abstract, index terms

This article presents a visual stepper for Common Lisp with simple backup feature. The stepper is based on Lieberman's approach for visual stepping. The stepper operates at the source level of programs and uses the GNU Emacs editor for source code processing. Problems encountered on implementing this stepper are described: read-syntax, macroexpansion, and source-localization. Especially, language features are discussed which affected our goal to make this tool portable across Common Lisp systems ...

12 Representing Java classes in a typed intermediate language
Christopher League, Zhong Shao, Valery Trifonov
September 1999 **ACM SIGPLAN Notices , Proceedings of the fourth ACM SIGPLAN international conference on Functional programming ICFP '99**, Volume 34 Issue 9
Publisher: ACM Press
Full text available: pdf(1.81 MB) Additional Information: full citation, abstract, references, citings, index terms

We propose a conservative extension of the polymorphic lambda calculus (*F*&omega;) as an intermediate language for compiling languages with name-based class and interface hierarchies. Our extension enriches standard *F*&omega; with recursive types, existential types, and row polymorphism, but only ordered records with no subtyping. Basing our language on *F*&omega; makes it also a suitable target for translation from other higher-order ...

13 Fast parallel implementation of lazy languages—the EQUALS experience
O. Kaser, S. Pawagi, C. R. Ramakrishnan, I. V. Ramakrishnan, R. C. Sekar
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming LFP '92**, Volume V Issue 1
Publisher: ACM Press
Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes EQUALS, a fast parallel implementation of a lazy functional language on a commercially available shared-memory parallel machine, the Sequent Symmetry. In contrast to previous implementations, we detect parallelism automatically by propagating exhaustive (normal form) demand. Another important difference between EQUALS and previous implementations is the use of reference counting for memory management instead of garbage collection. Our implementation shows that refere ...

14 From system F to typed assembly language
Greg Morrisett, David Walker, Karl Crary, Neal Glew
January 1998 **Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.39 MB) Additional Information: full citation, references, citings, index terms

15 Handling context-sensitive syntactic issues in the design of a front-end for a MATLAB compiler
Pramod G. Joisha, Abhay Kanhere, Prithviraj Banerjee, U. Nagaraj Shenoy, Alok Choudhary
March 2001 **ACM SIGAPL APL Quote Quad**, Volume 31 Issue 3
Publisher: ACM Press
Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references

In recent times, the MATLAB language has emerged as a popular alternative for programming in diverse application domains such as signal processing and meteorology. The language has a powerful array syntax with a large set of pre-defined operators and functions that operate on arrays or array sections, making it an ideal candidate for applications involving substantial array-based processing.Yet, for all the programming convenience that the language offers, designing a parser and scanner capable ...

Keywords: assignments, colon expressions, command-form function invocations, control constructs, matrices, single quote character, syntax analysis for MATLAB

16 On parallel processing of aggregate and scalar functions in object-relational DBMS
Michael Jaedicke, Bernhard Mitschang
June 1998 **ACM SIGMOD Record , Proceedings of the 1998 ACM SIGMOD international conference on Management of data SIGMOD '98**, Volume 27 Issue 2
Publisher: ACM Press
Full text available: pdf(1.43 MB) Additional Information: full citation, abstract, references, citings, index terms

Nowadays parallel object-relational DBMS are envisioned as the next great wave, but there is still a lack of efficient implementation concepts for some parts of the proposed functionality. Thus one of the current goals for parallel object-relational DBMS is to move towards higher performance. In this paper we develop a framework that allows to process user-defined functions with data parallelism. We will describe the class of partitionable functions that can be processed parallelly. We will ...

Keywords: aggregates, object-relational database systems, parallel query processing, user-defined functions

17 Agents, interactions, mobility and systems: Agent-based mobility add-in feature for Object Transaction Service (OTS)
Hoang Pham Huy, Simone Sedillot
March 2002 **Proceedings of the 2002 ACM symposium on Applied computing**
Publisher: ACM Press
Full text available: pdf(730.98 KB) Additional Information: full citation, abstract, references, index terms

Service session mobility is a new concept in the next telecom service generation. It allows users to move from one terminal in one network to another terminal in another network with service continuity. This concept requires migrating the current service execution related information, including service context, service data, etc, from one terminal to another. For services that execute within transactions, transaction context is a part of service context and must be migrated too. This paper prese ...

18 Demand-driven computation of interprocedural data flow
Evelyn Duesterwald, Rajiv Gupta, Mary Lou Soffa
January 1995 **Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.42 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a general framework for deriving demand-driven algorithms for interprocedural data flow analysis of imperative programs. The goal of demand-driven analysis is to reduce the time and/or space overhead of conventional exhaustive analysis by avoiding the collection of information that is not needed. In our framework, a demand for data flow information is modeled as a set of date flow queries. The derived demand-driven algorithms find responses to these queries through a par ...

19 A recovery mechanism for modular software
Flaviu Cristian
September 1979 **Proceedings of the 4th international conference on Software**

engineering
Publisher: IEEE Press
Full text available: pdf(897.65 KB) Additional Information: full citation, abstract, references, citings, index terms

When an exception occurs, the state of a system is damaged : further processing may cause additional exceptions. Under some hypotheses concerning the system structure we give an a priori estimate of the damage caused by exceptions. Based on this estimate we propose a recovery strategy and a recovery mechanism.

20 Federated database systems for managing distributed, heterogeneous, and autonomous databases
Amit P. Sheth, James A. Larson
September 1990 **ACM Computing Surveys (CSUR)**, Volume 22 Issue 3
Publisher: ACM Press
Full text available: pdf(5.02 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A federated database system (FDBS) is a collection of cooperating database systems that are autonomous and possibly heterogeneous. In this paper, we define a reference architecture for distributed database management systems from system and schema viewpoints and show how various FDBS architectures can be developed. We then define a methodology for developing one of the popular architectures of an FDBS. Finally, we discuss critical issues related to developing and operating an FDBS.

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **function invocation simd**

Found **136** of **193,448**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 136

Result page: **1** 2 3 4 5 6 7 next

Relevance scale

1 Automatic alignment of array data and processes to reduce communication time on DMPPs
Michael Philippsen
August 1995 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming PPOPP '95**, Volume 30 Issue 8
Publisher: ACM Press
Full text available: pdf(1.17 MB)
Additional Information: full citation, abstract, references, citings, index terms

This paper investigates the problem of aligning array data and processes in a distributed-memory implementation. We present complete algorithms for compile-time analysis, the necessary program restructuring, and subsequent code-generation, and discuss their complexity. We finally evaluate the practical usefulness by quantitative experiments.The technique presented analyzes complete programs, including branches, loops, and nested parallelism. Alignment is determined with respect t ...

2 LCM: memory system support for parallel language implementation
James R. Larus, Brad Richards, Guhan Viswanathan
November 1994 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the sixth international conference on Architectural support for programming languages and operating systems ASPLOS-VI**, Volume 29 , 28 Issue 11 , 5
Publisher: ACM Press
Full text available: pdf(1.27 MB)
Additional Information: full citation, abstract, references, citings, index terms

Higher-level parallel programming languages can be difficult to implement efficiently on parallel machines. This paper shows how a flexible, compiler-controlled memory system can help achieve good performance for language constructs that previously appeared too costly to be practical.Our compiler-controlled memory system is called Loosely Coherent Memory (LCM). It is an example of a larger class of Reconcilable Shared Memory (RSM) systems, which generalize the replication and mer ...

3 Function arrays
Robert Bernecky
June 1984 **ACM SIGAPL APL Quote Quad , Proceedings of the international conference on APL APL '84**, Volume 14 Issue 4
Publisher: ACM Press
Full text available: pdf(322.43 KB)
Additional Information: full citation, abstract, references, citings, index terms

A functional enclose operator, which returns the scalar representation of a function, is introduced. By providing a scalar representation of a function, functional enclose provides capabilities similar to those of SHARP APL Packages. When used in conjunction with the formatting function, functional enclose supplants (@@@@)cr. Apply is defined as a proper extension of the execute (@@@@) primitive function to the domain of enclosed function and array arguments. Apply, in conjunctio ...

4 Superfast parallel discrete event simulations
Albert G. Greenberg, Boris D. Lubachevsky, Isi Mitrani
April 1996 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 6 Issue 2
Publisher: ACM Press
Full text available: pdf(421.63 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Nonconventional parallel simulations methods are presented, wherein speed-ups are not limited by the number of simulated components. The methods capitalize on Chandy and Sherman's space-time relaxation paradigm, and incorporate fast algorithms for solving recurrences. Special attention is paid to implementing these algorithms on currently available massively parallel SIMD computers. As examples, "superfast" simulations for open and closed queuing networks and for the slotted ALO ...

Keywords: fixed-point computations, massively parallel computations, recurrences, relaxation, superlinear speedup, unbounded parallelism

5 On the relation between functional and data parallel programming languages
Per Hammarlund, Björn Lisper
July 1993 **Proceedings of the conference on Functional programming languages and computer architecture**
Publisher: ACM Press
Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

6 A low-power memory hierarchy for a fully programmable baseband processor
Wolfgang Raab, Hans-Martin Bluethgen, Ulrich Ramacher
June 2004 **Proceedings of the 3rd workshop on Memory performance issues: in conjunction with the 31st international symposium on computer architecture WMPI '04**
Publisher: ACM Press
Full text available: pdf(431.55 KB) Additional Information: full citation, abstract, references, index terms

Future terminals for wireless communication not only must support multiple standards but execute several of them concurrently. To meet these requirements, flexibility and ease of programming of integrated circuits for digital baseband processing are increasingly important criteria for the deployment of such devices, while power consumption and area of the devices remain as critical as in the past.The paper presents the architecture of a fully programmable system-on-chip for digital signal proces ...

Keywords: baseband processor, low-power memory, memory hierarchy, multi-tasked processor, task interleaving

7 Courses: Renderman for everyone
Rudy Cortes, Hal Bertram, Tal Lancaster, Dan Maas, Moritz Moeller, Heather Pritchett, Saty Raghavachary
July 2006 **Material presented at the ACM SIGGRAPH 2006 conference SIGGRAPH '06**
Publisher: ACM Press
Full text available: pdf(6.12 MB) Additional Information: full citation, abstract

An in-depth three-part course designed to expand knowledge of the RISpec. The first part is an introduction to RenderMan. The second is a detailed look into the RISpec. The third presents tips and tricks used in production.

8 GPGPU: general purpose computation on graphics hardware
David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn
August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**
Publisher: ACM Press
Full text available: pdf(63.03 MB) Additional Information: full citation, abstract, citings

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

9 A C compiler for a processor with a reconfigurable functional unit
Zhi Alex Ye, Nagaraj Shenoy, Prithviraj Baneijee
February 2000 **Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays**
Publisher: ACM Press
Full text available: pdf(477.60 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes a C compiler for a mixed Processor/FPGA architecture where the FPGA is a Reconfigurable Functional Unit (RFU). It presents three compilation techniques that can extract computations from applications to put into the RFU. The results show that large instruction sequences can be created and extracted by these techniques. An average speedup of 2.6 is achieved over a set of benchmarks.

10 Media and signal processing: Design and test of fixed-point multimedia co-processor for mobile applications
Ju-Ho Sohn, Jeong-Ho Woo, Jerald Yoo, Hoi-Jun Yoo
March 2006 **Proceedings of the conference on Design, automation and test in Europe: Designers' forum DATE '06**
Publisher: European Design and Automation Association
Full text available: pdf(202.32 KB) Additional Information: full citation, abstract, references

In this research, a fixed-point multimedia co-processor is designed and tested into an ARM-10 based mobile graphics processor for portable 2-D and 3-D multimedia applications. The fixed-point co-processor architecture with dual operations realizes advanced 3-D graphics algorithms and various streaming multimedia functions in a single hardware while consuming low power. The instruction-wise clock gating on fixed-point SIMD datapath allows fine-grained power control in application-specific manner. ...

11 Portable run-time support for dynamic object-oriented parallel processing
Andrew S. Grimshaw, Jon B. Weissman, W. Timothy Strayer
May 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 2
Publisher: ACM Press
Full text available: pdf(2.21 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Mentat is an object-oriented parallel processing system designed to simplify the task of writing portable parallel programs for parallel machines and workstation networks. The Mentat compiler and run-time system work together to automatically manage the communication and synchronization between objects. The run-time system marshals member function arguments, schedules objects on processors, and dynamically constructs and executes large-grain data dependence graphs. In this article we presen ...

Keywords: MIMD, dataflow, distributed memory, object-oriented, parallel processing

12 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4
Publisher: ACM Press
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

13 Compiling machine-independent parallel programs
Michael Philippsen, Ernst A. Heinz, Paul Lukowicz
August 1993 **ACM SIGPLAN Notices**, Volume 28 Issue 8
Publisher: ACM Press
Full text available: pdf(816.99 KB) Additional Information: full citation, abstract, citings, index terms

Initial evidence is presented that explicitly parallel, machine-independent programs can automatically be translated into parallel machine code that is competitive in performance with hand-written code.The programming language used is Modula-2*, an extension of Modula-2, which incorporates both data and control parallelism in a portable fashion. An optimizing compiler targeting MIMD, SIMD, and SISD machines translates Modula-2* into machine-dependent C code.The performance of the resulting code ...

14 Technical correspondence: Vector pascal reference manual
Paul Cockshott
June 2002 **ACM SIGPLAN Notices**, Volume 37 Issue 6
Publisher: ACM Press
Full text available: pdf(1.81 MB) Additional Information: full citation, references

15 Parallel programming with coordination structures
Steven Lucco, Oliver Sharp
January 1991 **Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.14 MB) Additional Information: full citation, references, citings, index terms

16 A parallel language and its compilation to multiprocessor machines or VLSI
Marina C. Chen
January 1986 **Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings

A language <b>Crystal</b> and its compiler for parallel programming is presented. The goal of <b>Crystal</b> is to help programmers in seeking efficient parallel implementations of an algorithm, and managing the complexity that might arise in dealing with hundreds of thousands of autonomous parallel processes. In <b>Crystal,</b> a program consists of a system of recursion equations and is interpreted as a parallel system. <b>Crystal</b> views a lar ...

17 Real-time shading
Marc Olano, Kurt Akeley, John C. Hart, Wolfgang Heidrich, Michael McCool, Jason L. Mitchell, Randi Rost
August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**
Publisher: ACM Press
Full text available: pdf(7.39 MB) Additional Information: full citation, abstract

Real-time procedural shading was once seen as a distant dream. When the first version of this course was offered four years ago, real-time shading was possible, but only with one-of-a-kind hardware or by combining the effects of tens to hundreds of rendering passes. Today, almost every new computer comes with graphics hardware capable of interactively executing shaders of thousands to tens of thousands of instructions. This course has been redesigned to address today's real-time shading capabili ...

18 Parallel programming with control abstraction
Lawrence A. Crowl, Thomas J. LeBlanc
May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3
Publisher: ACM Press
Full text available: pdf(3.68 MB) Additional Information: full citation, abstract, references, index terms, review

Parallel programming involves finding the potential parallelism in an application and mapping it to the architecture at hand. Since a typical application has more potential parallelism than any single architecture can exploit effectively, programmers usually limit their focus to the parallelism that the available control constructs express easily and that the given architecture exploits efficiently. This approach produces programs that exhibit much less parallelism that exists in the applic ...

Keywords: architectural adaptability, closures, control abstraction, data abstraction, early reply, multiprocessors, parallel programming languages, performance tuning

19 Dataflow machine architecture
Arthur H. Veen
December 1986 **ACM Computing Surveys (CSUR)**, Volume 18 Issue 4
Publisher: ACM Press
Full text available: pdf(3.19 MB) Additional Information: full citation, abstract, references, citings, index terms

Dataflow machines are programmable computers of which the hardware is optimized for fine-grain data-driven parallel computation. The principles and complications of data-driven execution are explained, as well as the advantages and costs of fine-grain parallelism. A general model for a dataflow machine is presented and the major design options are discussed. Most dataflow machines described in the literature are surveyed on the basis of this model and its associated technology. F ...

20 OOPAL: integrating array programming in object-oriented programming
Philippe Mougin, Stéphane Ducasse
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press
Full text available: pdf(158.90 KB) Additional Information: full citation, abstract, references, citings, index terms

Array programming shines in its ability to express computations at a high-level of abstraction, allowing one to manipulate and query whole *sets* of data at *once*. This paper presents the OPA model that enhances object-oriented programming with array programming features. The goal of OPA is to determine a minimum set of modifications that must be made to the traditional object model in order to take advantage of the possibilities of array programming. It is based on a minimal extensio ...

Keywords: array programming, f-script, high-level language, high-order messages, message pattern, smalltalk

Results 1 - 20 of 136 Result page: **1** 2 3 4 5 6 7 next

Sign in

Web Images Video News Maps more »

function invocation intermediate Search Advanced Search Preferences

Web Results **1** - **10** of about **654,000** for **function invocation intermediate**. (**0.22** seconds)

Optimizing compiler with static prediction of branch probability ...
generating **intermediate** code from source code of the program; ... determining a **function invocation** frequency cfreq(f), the **function invocation** frequency ...
www.patentstorm.us/patents/5655122-claims.html - 27k - Cached - Similar pages

Caml
The whole expression is just an **invocation** of the **function** ... user-defined **function** which gets the value of the element "x" and an **intermediate** result "r" ...
www.ocaml-programming.de/programming/page-1.html - 21k - Cached - Similar pages

CocoaDev: HigherOrderMessaging
Quite simply, -do is returning an **intermediate** and temporary ... In a functional language, your first order mechanism is **function invocation**, ...
www.cocoadev.com/index.pl?HigherOrderMessaging - 11k - Cached - Similar pages

[PDF] Semantics of **Functions** in Alloy
File Format: PDF/Adobe Acrobat - View as HTML
languages, Alloy **functions** are just typed macros, so an **invocation** of a **function** ... has the unfortunate side effect of not allowing unnamed **intermediate** ...
people.csail.mit.edu/dnj/teaching/6898/projects/sridharan.pdf - Similar pages

[PDF] Grail: a functional **intermediate** language for resource-bounded ...
File Format: PDF/Adobe Acrobat - View as HTML
Grail (Guaranteed resource allocation **intermediate** language) is a small ... of JVM bytecode with **function invocation** being implemented as a simple jump. ...
groups.inf.ed.ac.uk/mrg/publications/Grail-manual.pdf - Similar pages

Performance of CLR Integration
In the case of scalar-valued user-defined **functions**, this **function invocation** happens on a per-row basis. To minimize the cost of transitioning between SQL ...
msdn2.microsoft.com/en-us/ms131075.aspx - 20k - Cached - Similar pages

from-clause
If a **function**-name is specified, the **intermediate** result table is the set of rows returned by the table **function**. If a joined-table is specified, ...
publib.boulder.ibm.com/infocenter/iadthelp/
v6r0/topic/com.ibm.etools.iseries.langref2.doc/rbafzmst288.htm - 25k -
Cached - Similar pages

exit
If you do not specify a label (as described later), processing continues at the statement after the most recently run **function invocation** in the main ...
publib.boulder.ibm.com/infocenter/radhelp/
v6r0m1/topic/com.ibm.etools.egl.doc/topics/reglasm0748.html - 10k -
Cached - Similar pages
[More results from publib.boulder.ibm.com]

[PDF] Empirical Study of a Dataow Language on the CM-5
File Format: PDF/Adobe Acrobat - View as HTML
ments of a **function invocation**. In a conventional sequential language, a **function** is ... **function invocation**; the matching store is simply a means of ...
www.cs.cmu.edu/~seth/Empirical%20Study%20of%20a%20Dataflow%20Language.pdf -
Similar pages

Guide to the use of Character Sets in Europe
The coded representation of such a control **function** is by means of an escape sequence in which the first **Intermediate** Byte has value 25. ...
anubis.dkuug.dk/CEN/TC304/guide/gucsch10.htm - 17k - Cached - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

Get organized for the new year with Google Desktop.

function invocation intermediate Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Sign in

Web Images Video News Maps **more »**

function invocation simd Search Advanced Search Preferences

Web Results **1** - **10** of about **31,200** for **function invocation simd**. (**0.35** seconds)

[PDF] Research on programming languages for massively parallel ...
File Format: PDF/Adobe Acrobat
from **SIMD** type parallel language which has been developed by the group of Toyohashi ...
Function invocation is done at all active virtual processors. ...
ieeexplore.ieee.org/iel2/2965/8408/00367134.pdf?arnumber=367134 - Similar pages

[PDF] Dynamic, object-oriented parallel processing - IEEE Parallel ...
File Format: PDF/Adobe Acrobat
(**SIMD**). code. Conse-. quently, many opportunities for parallelism are lost. ... LMentat object member **function invocation** (as on line ...
ieeexplore.ieee.org/iel4/88/5729/00218174.pdf?arnumber=218174 - Similar pages

C++ Futures: Lambda **Functions** | The Register
Lambda **functions** or Lambdas in C++ are one of the more interesting things to look forward to ... PPU or **SIMD** architectures) can generate much better code. ...
www.regdeveloper.co.uk/2006/09/22/cplusplus-lambda-future/ - 30k -
Cached - Similar pages

Description of compiler flags for Intel C++ Compiler 9.0 and 9.1 ...
P = Intel Pentium 4 processor with Streaming **SIMD** 3 (SSE3) support. ... it determines that the stack pointer is fixed throughout the **function invocation**. ...
www.spec.org/cpu/flags/FSC-20060609-Intel91-90-PGI60-Pathscale23.txt - 62k -
Cached - Similar pages

Copyright 2003, 2004, 2005 PathScale, Inc. All Rights Reserved ...
-LNO:**simd**=(0|1|2) This option enables or disables inner loop vectorization. ... that the stack pointer is fixed throughout the **function invocation**. ...
www.spec.org/cpu/flags/IBM-20051213-ps2.2.1.txt - 27k - Cached - Similar pages
[More results from www.spec.org]

ACESgrid.org: Itanium2 nodes and IA64 software
It provides synchronous/asynchronous remote **function invocation** facilities, ... The scout program provides a **SIMD**-style remote UNIX shell environment. ...
acesgrid.org/itanium2_nodes_and_ia64_software.html - 24k - Cached - Similar pages

IBM JRD 49-4/5 | Introduction to the Cell multiprocessor
The Cell processor uses a **SIMD** organization in the vector unit on the PPE and ... in a read-only section along with a small remote **function invocation** stub. ...
www.research.ibm.com/journal/rd/494/kahle.html - 81k - Cached - Similar pages

Stream processing - Wikipedia, the free encyclopedia
While stream processing is a branch of **SIMD**/MIMD processing, ... identifying all the values which are short-lived to a single kernel **invocation**. ...
en.wikipedia.org/wiki/Stream_processing - 55k - Cached - Similar pages

[PS] The Good News About Dynamic Object-Oriented Parallel Processing ...
File Format: Adobe PostScript - View as Text
SIMD. Consequently, many opportunities for parallelism are lost. ... Since state must be maintained, each member **function invocation** on a persistent ...
www.cs.virginia.edu/~techrep/CS-92-41.ps.Z - Similar pages

What's new in version 7
The **SIMD** (Single Instruction, Multiple Data) instruction set enables higher ... The compiler **invocation** xlc++ has been added for portability among all ...

publib.boulder.ibm.com/infocenter/lnxpcomp/v7v91/topic/com.ibm.vacpp7l.doc/getstart/overview/new_features.htm - 30k - Cached - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

Get organized for the new year with Google Desktop.

function invocation simd Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Home | Login | Logout | Access Information | Alerts | Sitemap | Help

Welcome United States Patent and Trademark Office

☐ Search Results

BROWSE SEARCH IEEE XPLORE GUIDE SUPPORT

Results for "**((simd compiler)<in>metadata)**"
Your search matched **2** of **1450046** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

e-mail printer friendly

» Search Options

View Session History

New Search

Modify Search

((simd compiler)<in>metadata) Search

☐ Check to search only within this results set

Display Format: [✓ Citation] Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

view selected items **Select All** **Deselect All**

☐ 1. **Context optimization for SIMD execution**
Kennedy, K.; Roth, G.;
Scalable High-Performance Computing Conference, 1994. Proceedings of the
23-25 May 1994 Page(s):445 - 453
Digital Object Identifier 10.1109/SHPCC.1994.296677
AbstractPlus | Full Text: PDF(692 KB) **IEEE CNF**
Rights and Permissions

☐ 2. **Design and performance of an optimizing SIMD compiler**
Fisher, A.L.; Leon, J.; Highnam, P.T.;
Frontiers of Massively Parallel Computation, 1990. Proceedings., 3rd Symposium on the
8-10 Oct. 1990 Page(s):507 - 510
Digital Object Identifier 10.1109/FMPC.1990.89504
AbstractPlus | Full Text: PDF(292 KB) **IEEE CNF**
Rights and Permissions

Help Contact Us Privacy & Security IEEE.org

indexed by Inspec